

(19) 日本国特許庁 (J P)

公開特許公報 (A)

(11) 特許出願公開番号

特開平8-123651

(43) 公開日 平成8年(1996)5月17日

(51) Int. Cl.
G06F 3/14

識別記号
350 A

F I

審査請求 未請求 請求項の数 2 O L (全8頁)

(21) 出願番号 特願平6-264224

(22) 出願日 平成6年(1994)10月27日

(71) 出願人 000005234

富士電機株式会社

神奈川県川崎市川崎区田辺新田1番1号

(71) 出願人 000237156

富士ファコム制御株式会社

東京都日野市富士町1番地

(72) 発明者 澤田 孝雄

東京都日野市富士町1番地 富士ファコム
制御株式会社内

(74) 代理人 弁理士 大菅 義之

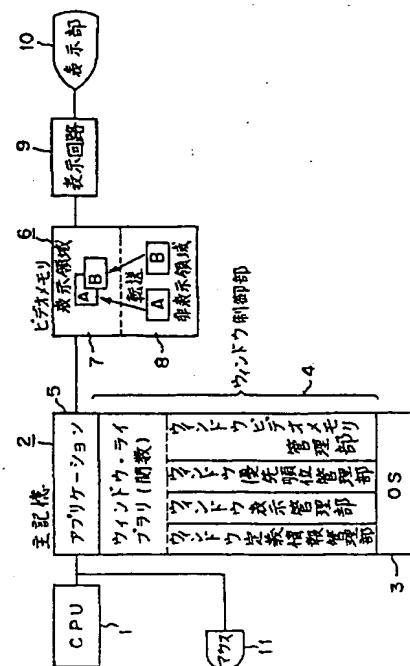
(54) 【発明の名称】マルチウィンドウディスプレイ装置

(57) 【要約】

【目的】 ウィンドウ表示更新時間を短縮することにより、ウィンドウ表示を更新する際のちらつきが生じないようにする。

【構成】 アプリケーション等からウィンドウの表示要求があった場合、ウィンドウ制御部4は、まず、ウィンドウの定義情報をウィンドウ定義情報管理部などに登録し、ウィンドウ全体の表示データを格納するためのウィンドウワークバッファをビデオメモリ6の非表示領域8に獲得する。次に、獲得されたウィンドウワークバッファに対してウィンドウ内に表示させる文字・図形等の描画を行う。そして、ウィンドウワークバッファに描画された表示データを、ウィンドウ定義情報等に基づいて、ビデオメモリ6の表示領域7に矩形転送し、表示部10の画面上に表示させる。

本発明の実施例のシステム構成図



【特許請求の範囲】

【請求項1】表示画面に対応する表示データを格納する表示領域と表示画面に表示されていない表示データを格納する非表示領域の2つの領域をもつ表示データ格納手段と、

前記表示領域に格納されている表示データを読み出して表示する表示手段と、

前記非表示領域から前記表示領域へ表示データを転送する制御手段とを備え、

前記非表示領域に、前記表示手段に表示させるすべてのウィンドウの全表示データを格納させることを特徴とするマルチウィンドウディスプレイ装置。

【請求項2】前記制御手段は、前記表示手段に表示させるウィンドウ間の重なり判定を行い、該判定により得られるウィンドウの可視領域の表示データのみを前記非表示領域から前記表示領域へ矩形転送することを特徴とする請求項1記載のマルチウィンドウディスプレイ装置。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、マルチウィンドウ表示が可能なディスプレイ装置に関する。

【0002】

【従来の技術】複数のウィンドウを画面上に表示させるマルチウィンドウディスプレイ装置では、従来、ウィンドウの表示形態としてウィンドウ同士が重なり合うことを許可するオーバーラップ型ウィンドウシステムを採用したものが多く使われている。また、画面上の各ウィンドウ内に表示される文字・図形等（画面情報）は描画コマンド（例えば、線・円等の描画コマンド）としてメモリ（主記憶）に保存されており、装置内のCPU（中央処理装置）が、メモリから該描画コマンドを読み出し、読み出した描画コマンド実行することで、ウィンドウ内に文字・図形等の描画を行っていた。

【0003】図9は、従来のマルチウィンドウディスプレイ装置の構成ブロック図である。図9のような構成において、例えば、磁気ディスク装置等から主記憶72上に読み込まれたアプリケーション75がウィンドウシステム74に対してウィンドウの表示要求を行った場合、ウィンドウシステム74は、まず、該表示要求によってこれから表示部78の画面上に表示させるウィンドウと、例えば、前記アプリケーションからの該表示要求以前の表示要求もしくは他のアプリケーションからの表示要求などにより既に画面上に表示されているウィンドウとの間で重なり制御を行う。そして、ウィンドウの画面情報である描画コマンドを主記憶72の特に図示していない領域から読み出して、該描画コマンドを実行することにより表示データを生成し、該表示データをビデオメモリ76に書き込む。そして、それと同時にウィンドウの画面情報である描画コマンドを主記憶72の特に図示していない領域に記憶（保存）する。この保存された描

画コマンドは、後述するように、ウィンドウの表示形態が変化した場合に使用される。ビデオメモリ76に書き込まれた表示データは、表示回路77を介して表示部78の画面に表示される。

【0004】複数のウィンドウが互いに重なり合って表示された状態で、利用者がマウス79などを操作してウィンドウの移動や消去等を行うと、ウィンドウ間の重なり状態が変化して今まで表示されていなかったウィンドウの一部または全部を表示する必要が生じる。この場合は、画面を更新するのに、一度隠されていた領域を画面の背景色でクリアし、主記憶72に記憶されている前記描画コマンドを読み出して、該描画コマンドを実行することによりウィンドウの表示データを生成し、該表示データをビデオメモリ76に書き込むことにより、再描画を行っていた。

【0005】

【発明が解決しようとする課題】上述したように、従来のマルチウィンドウディスプレイ装置では、ウィンドウの移動・消去等によって、ウィンドウ間の重なり状態が変化し、ウィンドウの今まで隠されていた領域を表示させる必要が生じた場合、一度隠されていた領域を画面の背景色でクリアしてから、主記憶上に保存されている描画コマンドを使用して文字・図形等を再度描画することで画面修復を行っていた。そのため、描画コマンドの読み出しや、該描画コマンドの実行による表示データの生成等の再描画に要する処理時間が長くかかり、ウィンドウの重なり状態が変化した際に、画面上でちらつきが生じるという欠点があった。

【0006】本発明の課題は、ウィンドウ表示更新時間を短縮することにより、ちらつきが生じないマルチウィンドウディスプレイ装置を提供することである。

【0007】

【課題を解決するための手段】請求項1記載の発明は、図1に示すように以下の各手段を備える。表示データ格納手段100は、表示画面に表示可能な表示データを格納する記憶装置であり、更に、表示画面に対応する表示データを格納する表示領域101と表示画面に表示されていない表示データを格納する非表示領域102の2つの領域を持つ。

【0008】表示手段103は、前記表示領域101に格納されている表示データを読み出して表示画面に表示する。制御手段104は、前記非表示領域102から前記表示領域101へ表示データを転送する。

【0009】そして、表示画面上に表示させるすべてのウィンドウの全表示データを前記非表示領域102に格納せしめるようにする。また、前記制御手段104が、表示画面上に表示させるウィンドウ間の重なり判定を行い、該判定によって得られるウィンドウの可視領域の表示データのみを前記非表示領域102から前記表示領域101へ矩形転送するようにしてもよい。

【0010】

【作用】ウィンドウの重なり状態などが変化して、表示画面の更新処理を行う場合、制御手段104は、表示データ格納手段100の非表示領域102に格納されているウィンドウの表示データを表示データ格納手段100の表示領域101に転送する。こうすることによって、表示画面を更新する際にウィンドウの表示データを描画コマンドを実行することで再描画する必要がなくなり、高速な画面更新ができるようになる。

【0011】つまり、ウィンドウの全表示データを非表示領域102に格納するようにしているので、表示画面上では他のウィンドウに隠されている部分の表示データ、すなわち、表示領域101には存在していない表示データも、非表示領域102には存在している。それゆえ、ウィンドウの重なり状態などが変化して、今まで隠されていた部分の表示データが必要になった場合でも、該表示データを描画コマンドを実行して新たに生成する必要はなく、非表示領域102から表示領域101へ表示データを転送するだけで、画面更新が行える。

【0012】また、制御手段104が、ウィンドウ間の重なり判定を行い、該判定によって得られるウィンドウの可視領域の表示データのみを前記非表示領域102から前記表示領域101へ矩形転送するようにすれば、データ転送量を少なくすることができます。

【0013】

【実施例】以下、本発明の実施例について、図面を参照にしながら詳細に説明する。図2は本実施例のマルチウィンドウディスプレイ装置のシステム構成を示すブロック図である。

【0014】図2に示すように、本実施例は、CPU(中央処理装置)1と、主記憶2と、ビデオメモリ6と、表示回路9と、表示部10と、及びマウス11から構成される。

【0015】CPU1は、オペレーティング・システム(OS)3や、ウィンドウ・ライブラリ(関数)や各種アプリケーション5を実行することにより本実施例の装置全体を制御する。

【0016】主記憶2は、CPU1が実行するオペレーティング・システム(OS)3、ウィンドウ制御部4、及び各種アプリケーション5等を格納する記憶装置である。ウィンドウ制御部4は、ウィンドウ・ライブラリと、ウィンドウ定義情報管理部と、ウィンドウ表示管理部と、ウィンドウ優先順位管理部、及びウィンドウビデオメモリ管理部から構成される。

【0017】ウィンドウ・ライブラリは、ウィンドウ表示などウィンドウ制御に関連する処理を行うプログラム群であり、CPU1によって実行される。ウィンドウ定義情報管理部は、ウィンドウの定義情報や描画属性の管理を行う。ウィンドウの定義情報としては、例えば、ウィンドウ識別子番号、親ウィンドウ識別子番号、ウィン

ドウ生成サイズ(ウィンドウ本体の大きさ)、ウィンドウ表示サイズ(ウィンドウ本体を実際に表示部10の画面上に表示する際の大きさ)、ウィンドウ表示位置(ウィンドウを表示する表示部10の画面上の座標)、ウィンドウ内表示開始位置、ウィンドウ背景色等がある。

【0018】ウィンドウ表示管理部は、表示部10の画面に表示されているウィンドウの矩形領域座標の情報、ウィンドウ同士の重なり関係を管理する。ウィンドウ優先順位管理部は、ウィンドウを重ねて表示する際の上下関係を管理する。

【0019】ウィンドウビデオメモリ管理部は、ウィンドウ生成時にビデオメモリ6の非表示領域8に獲得されるウィンドウワークバッファの使用アドレス・サイズを管理する。

【0020】ビデオメモリ6は、表示部10に表示することが可能な表示データが格納される記憶装置である。本実施例では、ビデオメモリ6は、表示部10の画面と対応した表示領域7とそれ以外の非表示領域8との2つの領域に分けられる。表示領域7に書き込まれた表示データは後述する表示回路9によって表示部10の画面上に表示される。これに対して、非表示領域8に書き込まれた表示データは、表示部10画面上に表示されることはない。CPU1は、表示データの読み書きをこの二つの領域に対して同様に行える。なお、本実施例では、表示領域7に対して非表示領域8は3倍の記憶容量を持っている。

【0021】表示回路9は、ビデオメモリ6の表示領域7に書き込まれた表示データを適宜読み出して、その表示イメージを表示部10に表示させる制御回路である。

30 なお、表示回路9は、ビデオメモリ6の非表示領域8に書き込まれた表示データを読み出すことはできない。

【0022】表示部10は、複数のウィンドウなどを表示する表示装置であり、例えば、CRT(cathode ray tube)ディスプレイや液晶ディスプレイなどからなる。マウス11は、ポインティングデバイスであり、例えば、表示部10に表示されるウィンドウの移動や変形等の操作を指示するのに用いられる。

【0023】次に、本実施例のマルチウィンドウディスプレイ装置が行う処理の説明する。本実施例では、ウィンドウに関する処理として、以下のようないくつかの処理が行われる。アプリケーション等からウィンドウの表示要求があった場合、ウィンドウ制御部4は、まず、ウィンドウ生成処理として、ウィンドウの定義情報をウィンドウ定義情報管理部などに登録し、ウィンドウ全体の表示データを格納するためのウィンドウワークバッファをビデオメモリ6の非表示領域8に獲得する。該ウィンドウワークバッファは、例えば、後述するウィンドウA、B等の、ウィンドウ全体の表示データを格納する領域である。次に、画面情報描画処理として、ビデオメモリ6の非表示領域8に獲得されたウィンドウワークバッファに対して

5
ウィンドウ内に表示させる文字・図形等の描画を行う。そして、ウィンドウ表示処理として、ウィンドウワーカバッファに描画された表示データを、ウィンドウ定義情報等に基づいて、ビデオメモリ6の表示領域7に矩形転送し、表示部10の画面上に表示させる。

【0024】また、このようにして表示されたウィンドウに対して、利用者のマウス11の操作により、ウィンドウの移動が指示されると、ウィンドウ制御部4は、表示部10の画面上での該ウィンドウの表示位置を変更し、その後、後述するウィンドウ表示処理により画面の更新を行う。また、同様に、マウス11の操作等により、ウィンドウの変形が指示されると、ウィンドウの表示されている大きさを指示に従って変更し、その後、後述するウィンドウ表示処理により画面の更新を行う。

【0025】さらに、アプリケーションの終了などによりウィンドウを削除する場合、ウィンドウ制御部4は、表示部10の画面上に表示されている該ウィンドウを消去し、該ウィンドウの定義情報の削除及びウィンドウワーカバッファの解放を行い、その後、後述するウィンドウ表示処理により画面の更新を行う。

【0026】次に、本実施例のマルチウィンドウディスプレイ装置の動作の詳細を、ウィンドウの表示と消去を例として説明を行う。まず、ウィンドウ表示時の本実施例の動作を説明する。ここでのウィンドウ表示とは、非表示領域に描画されているウィンドウ本体を表示領域である画面上に矩形転送して可視状態にすることをいう。

【0027】図3は、ウィンドウA、B、Cが生成され、その内ウィンドウA、Bは表示部10の画面上に表示済みである時のビデオメモリ3の状態を模擬的に示している。従って、ウィンドウCは、初期状態ではウィンドウの生成はされているが、表示部10の画面上には表示されていない。このとき、現在のウィンドウの重なり状態に合わせて各ウィンドウは、図3に示されるように、可視矩形A11、B11、B12及び不可視矩形B13に分割されている。なお、このときの表示優先度はA>B>Cの順である。

【0028】図3に示す状態より、ウィンドウCを画面上にウィンドウ表示する場合、まず、表示優先度の低い順に2つのウィンドウ間で重なり状態の判定を行う。そして、該判定を画面上に表示するウィンドウの可視矩形全ての組み合わせについて行う。以下に該判定の際に行われる矩形分割方法の詳細を図4を参照しながら説明する。

【0029】(a) まず、ウィンドウCとウィンドウB間で重なり判定を行う。その結果、図4(a)に示すようにウィンドウCは不可視矩形C11と可視矩形C12に分割される。ウィンドウBは全ての領域がそのまま可視矩形B11となる。

【0030】(b) 次に、ウィンドウCの可視矩形C12とウィンドウA間で重なり状態判定を行う。その結果、

10
図4(b)に示すように可視矩形C12が更に不可視矩形C122と可視矩形C121、C123に分割される。ウィンドウAは全ての領域がそのまま可視矩形A11となる。

【0031】(c) 最後に、ウィンドウB(可視矩形B11)とウィンドウA(可視矩形A11)間で重なり状態判定を行う。その結果、図4(c)に示すように可視矩形B11は不可視矩形B113と可視矩形B111、B112に分割される。可視矩形A11はそのままの状態である。

【0032】前記(a)～(c)の処理の結果得られた、ウィンドウCの可視矩形C121、C123を表示領域に矩形転送すると、図5に示す状態になる。なお、この場合、ウィンドウA、Bの表示状態は変化しないので、ビデオメモリ6の表示領域7に格納されているウィンドウA、Bの表示データに変化はない。このようにして、ウィンドウCの表示処理が行われる。

【0033】次に、ウィンドウの消去時の本実施例の動作を説明する。ウィンドウの消去とは、画面上に表示さ

20
れていますのウィンドウを不可視状態にするものである。図5に示す状態において、ウィンドウBを消去する場合、まず、ウィンドウBの可視領域を画面の背景色でクリアすることでウィンドウBを消去し、次に、ウィンドウBよりも表示優先度の低いウィンドウの不可視矩形とその他のウィンドウ間で重なり状態判定を行う。そして、ウィンドウBが隠していた領域が他のウィンドウにより再び隠されるか否かの判定を行い、表示されるべき領域を算出する。以下に再表示領域算出方法を図6を参照しながら説明する。

30
30
【0034】まず、ウィンドウBに隠されていたウィンドウCの不可視矩形C11とウィンドウA(可視矩形A11)との間で重なり状態判定を行う。その結果、図6(a)に示すように、不可視矩形C11は矩形C111と矩形C112に分割される。矩形C111は可視矩形、C112は再び不可視矩形となる。

40
40
【0035】このようにして得られたウィンドウCの再表示領域C111を非表示領域より矩形転送すると表示画面が更新される。さらに、前述したウィンドウ表示の場合と同様にして画面更新後の各ウィンドウを可視矩形と不可視矩形に分割する。そのため、ウィンドウB消去後のウィンドウCとウィンドウA間の重なり状態判定を行う。その結果、図6(b)に示すように、ウィンドウCは不可視矩形C12と可視矩形C11、C13に分割される。ウィンドウAは全ての領域がそのまま不可視矩形A11となる。

【0036】上述したように、ウィンドウBの可視領域を画面の背景色でクリアし、ウィンドウCの再表示領域C111を非表示領域より表示領域へ矩形転送すると表示画面が更新される。さらに、ウィンドウB消去後の表示ウィンドウ間の重なり状態を判定して矩形分割を行

と図 7 に示す状態となる。このようにして、ウィンドウ B の消去処理が行われる。

【0037】図 8 は、上記説明したウィンドウ表示(更新)処理の動作を説明するフローチャートである。なお、以下に示すフローチャートの説明中、S 1、S 2、……は処理手順(ステップ)の番号を示す。

【0038】アプリケーションからのウィンドウ表示要求や、ウィンドウの移動・削除等に伴うウィンドウ表示要求を受け取ると、CPU 1 は、まず、表示優先度の低いウィンドウから順々に 2 つのウィンドウを選択して、

それらが重なっているか否かの判定を行う(S 1、S 2)。該判定の結果、ウィンドウが重なっていた場合

(S 2、YES)、該ウィンドウの既存可視矩形間での重なり判定を行う(S 3)。該判定の結果、既存可視矩形間で重なりがあった場合(S 3、YES)、上述したような可視矩形分割処理を行う(S 4)。また、該判定の結果、既存可視矩形間で重なりがなかった場合(S 3、NO)は、該可視矩形分割処理を行わず、次の処理に進む。

【0039】そして、現在処理中のウィンドウの既存の可視矩形すべてに対して処理が終了したか否かを判定する(S 5)。該判定の結果、既存の可視矩形でまだ処理していないものがあれば(S 5、NO)、上記ステップ S 3～S 4 を繰り返す。また、該判定の結果、既存の可視矩形すべてに対して処理が終了していれば(S 5、YES)、さらに、画面に表示すべきウィンドウの全組み合わせに対して処理が終了したか否かを判定する(S 6)。なお、上記ステップ S 2 で、選択されたウィンドウが重なっていないと判定した場合(S 2、NO)も、ステップ S 6 の処理に進む。

【0040】該判定の結果、表示すべきウィンドウの組み合わせで、まだ処理していない組み合わせがあれば(S 6、NO)、上記ステップ S 1～S 5 を繰り返す。また、該判定の結果、画面に表示すべきウィンドウの全組み合わせに対して処理が終了していれば(S 6、YES)、上記処理によって求められた、表示するウィンドウの可視領域を非表示領域から表示領域に矩形転送を行う(S 7)。

【0041】なお、上記実施例において、ウィンドウの数が増えて、ビデオメモリ 6 の非表示領域 8 が足りなくなった場合は、上記ウィンドウ生成処理時に、ビデオメモリ 6 の非表示領域 8 の再構成処理を行う。再構成処理とは、ウィンドウの生成及び消去が繰り返されることに

よって、飛び飛びになっている非表示領域 8 の空き領域を、まとめて、1 つの空き領域にする処理である。

【0042】さらに、ビデオメモリ 6 の非表示領域 8 の再構成処理を行っても、ビデオメモリ 6 が足りない場合には、他のウィンドウを削除しない限り、新規にウィンドウを作成できないので、その旨を利用者に通知するメッセージ等を表示部 10 の画面上に表示する。

【0043】

【発明の効果】以上、詳細に説明したように、本発明によれば、ウィンドウの表示あるいは画面表示更新処理を、再描画方式ではなく矩形転送(ビデオメモリ間のブロック転送処理)による再表示方式で行うため、文字・図形等の描画処理時間を省くことができ、表示更新時間の短縮が図られて高速表示が可能となる。このため、ウィンドウの重なり状態が変化した際に、ちらつきが生じることもなくなる。

【図面の簡単な説明】

【図 1】本発明の原理図である。

【図 2】本発明の実施例のシステム構成図である。

【図 3】実施例のビデオメモリの状態を示す図である。

【図 4】ウィンドウの矩形分割処理を説明する図である。

【図 5】実施例のビデオメモリの状態を示す図である。

【図 6】ウィンドウの矩形分割処理を説明する図である。

【図 7】実施例のビデオメモリの状態を示す図である。

【図 8】ウィンドウ表示処理の動作を説明するフローチャートである。

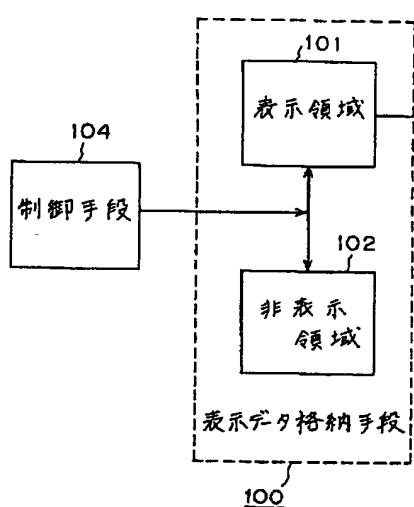
【図 9】従来のマルチウィンドウディスプレイ装置の構成ブロック図である。

【符号の説明】

1	CPU
2	主記憶
3	OS
4	ウィンドウ制御部
5	アプリケーション
6	ビデオメモリ
7	表示領域
8	非表示領域
9	表示回路
10	表示部
11	マウス

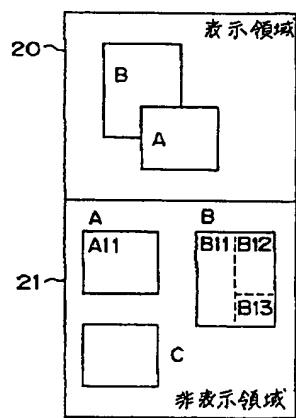
【図1】

本発明の原理図



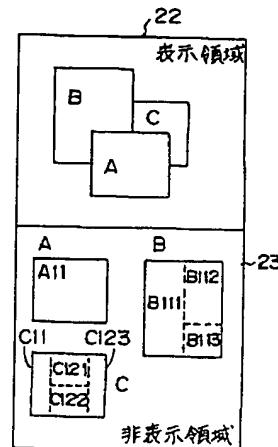
【図3】

実施例のビデオメモリの状態を示す図



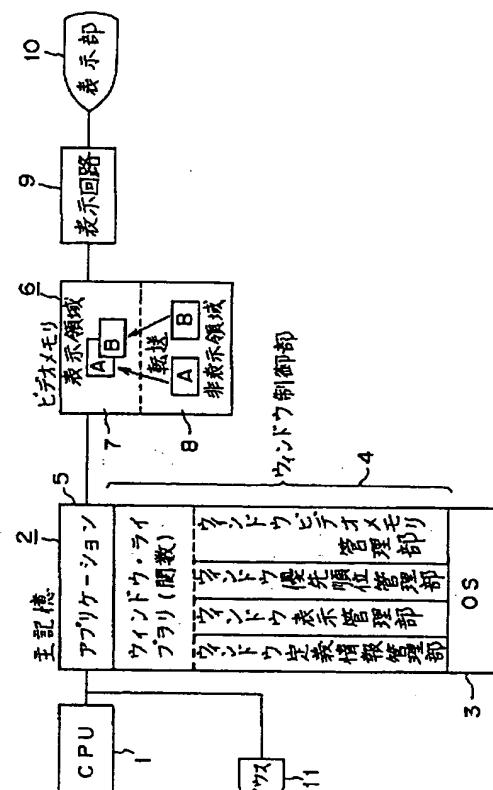
初期状態

実施例のビデオメモリの状態を示す図 実施例のビデオメモリの状態を示す図

初期状態より
ウィンドウ C を表示

【図2】

本発明の実施例のシステム構成図



【図7】

実施例のビデオメモリの状態を示す図

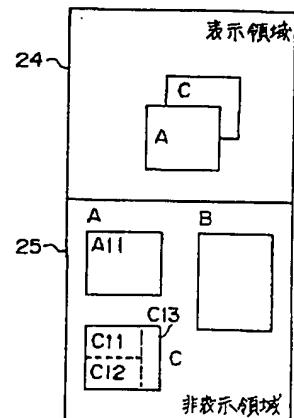
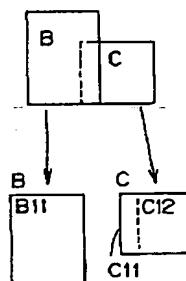


図5の状態よりウンドウ C を消去

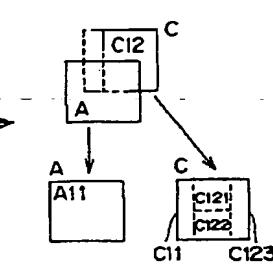
【図4】

ウインドウの矩形分割処理を説明する図

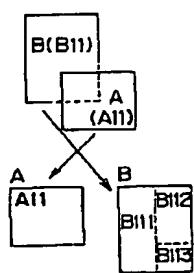
(a) ウィンドウBとCの重なり状態



(b) ウィンドウCの可視矩形C12とウィンドウAの重なり状態



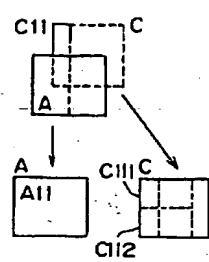
(c) ウィンドウA(A11)とB(B11)の重なり状態



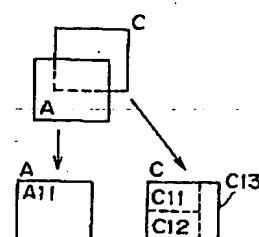
【図6】

ウインドウの矩形分割処理を説明する図

(a) ウィンドウCの不可視矩形C11とウィンドウAの重なり状態

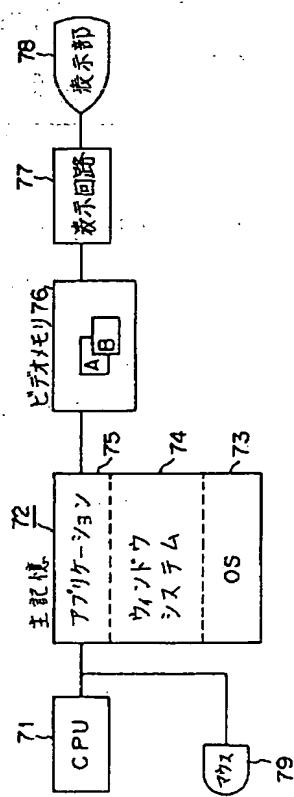


(b) ウィンドウCとウィンドウAの重なり状態



【図9】

従来のマルチウィンドウディスプレイ装置の構成ブロック図



【図8】

ウインドウ表示処理の動作を説明するフローチャート

